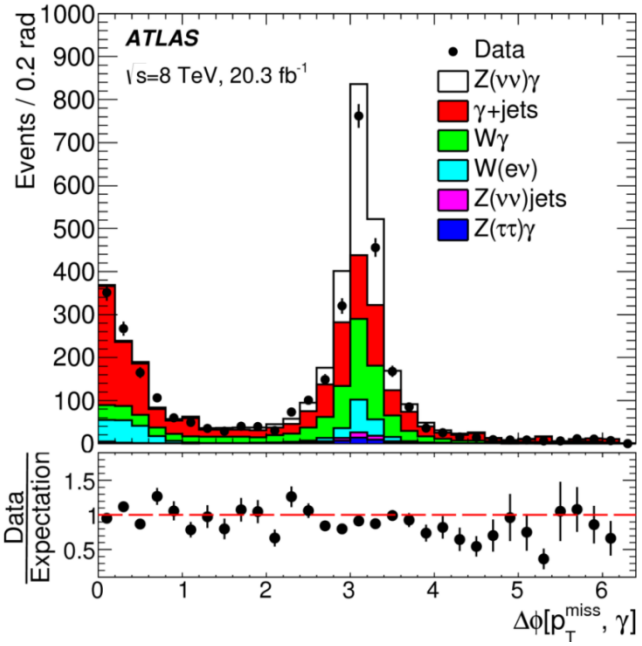
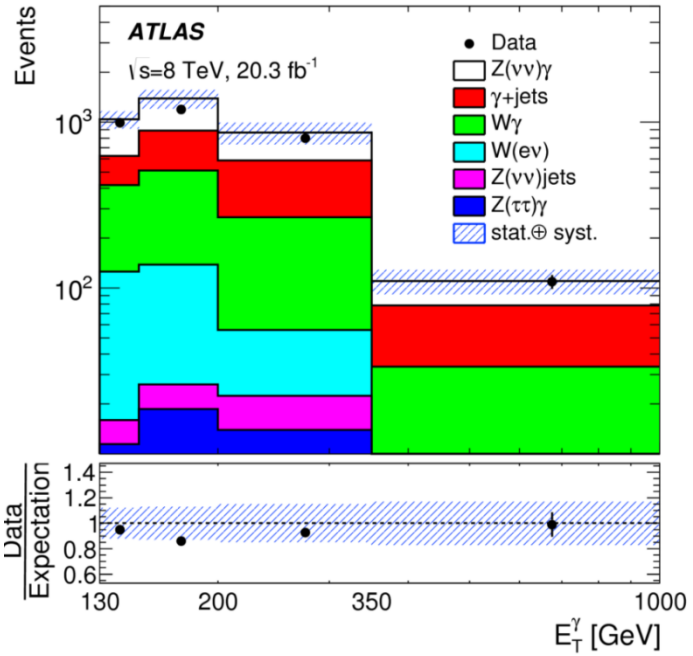
Лекции. Практические занятия

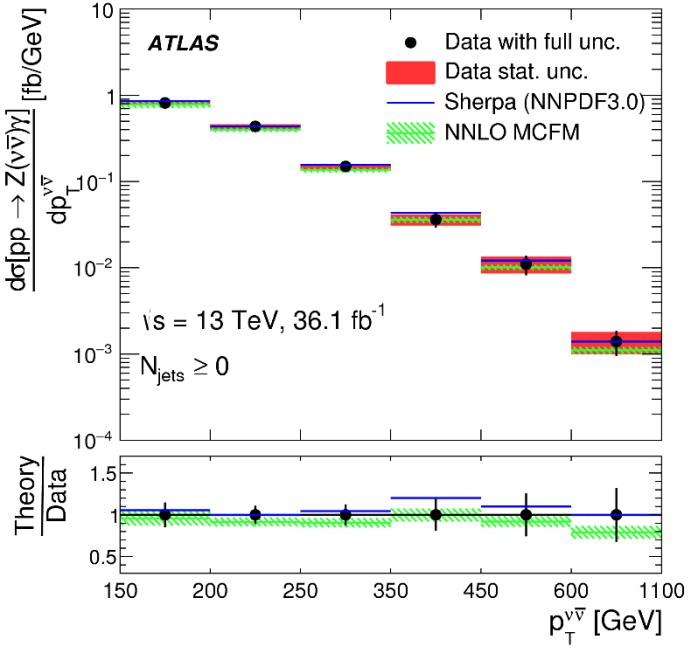
Солдатов Е.Ю.

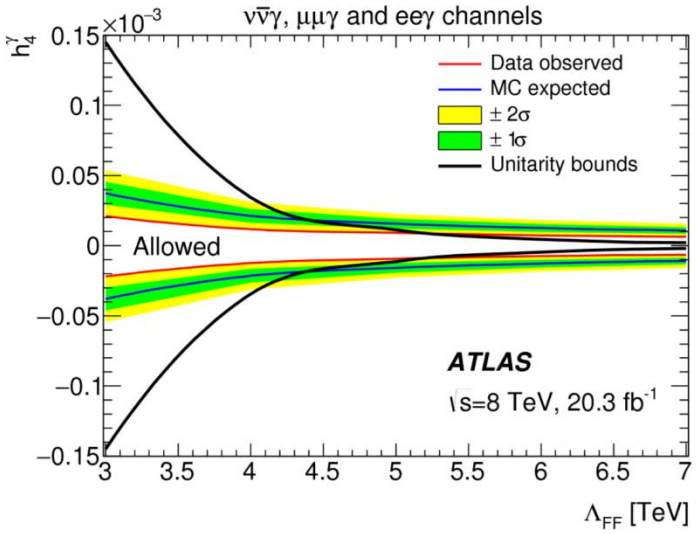
2025 г.

ГИСТОГРАММЫ









ГИСТОГРАММЫ

- ROOT поддерживает гистограммы, в которых не более 3 измерений. Мы начнём с одномерных гистограмм.
- Гистограмма – это наглядное изображение распределения случайной величины при условии ограниченности статистики выборки

- Базовый класс: TH1 (TH2, TH3)
- Основные используемые классы:

TH1C, TH2C и TH3C резервируют по 1 байту на бин (максимальное содержимое 1 бина = 255)

TH1I, TH2I и TH3I резервируют одну переменную типа int на бин (максимальное содержимое бина = 2 147 483 647).

TH1F, TH2F и TH3F резервируют одну переменную типа float на бин (максимальная точность бина, значащих цифр = 7 digits).

TH1D, TH2D и TH3D резервируют одну переменную типа double на бин (максимальная точность бина, значащих цифр = 14 digits).

Ref. manual: https://root.cern.ch/doc/master/classTH1.html

ГИСТОГРАММЫ

- Конструкторы:

```
TH1F  *h1  = new TH1F("HistName", "Histogram title", Nbins, xmin, xmax)

double xarray[Nbins+1]={xmin,…,xmax};
TH1F *h1  = new TH1F("HistName", "Histogram title", Nbins, xarray)
```

Создается гистограмма (объект класса TH1F) и указатель h1 на этот объект.

- Параметры:

• HistName – имя гистограммы, без пробелов
• Histogram title – заголовок гистограммы
Заголовок гистограммы может сразу включать и заголовки осей, например: “Histogram title; x [mm]; dN/dx”
• Nbins – число бинов в гистограмме (целая величина)
• xmin, xmax – диапазон изменения гистограммируемой величины
• xarray – массив границ бинов (для неравномерного биннирования)

ГИСТОГРАММЫ

Наиболее важные методы:

Заполнение и манипуляция данными внутри гистограммы

- h1->Fill(value, weight) // weight – необязательный параметр.

h1->SetBinContent(bin, value) h1->SetBinError(bin, error)

h1->GetBinContent(bin) h1->GetBinError(bin)

Изменение предустановленных характеристик:

h1->SetTitle("title")

h1->Rebin(n_m_bins, new_title, xarray) // xarray - необязательный

Изменение внешнего вида

h1->SetFillColor(color) h1->SetLineWidth(n_of_pt)

h1->SetMarkerStyle(n_marker)

Подписать оси гистограмм:

h1->GetXaxis()->SetTitle("Xaxis title"); h1->GetYaxis()->SetTitle("Yaxis title");

Получить интеграл гистограммы:

h1->Integral()

Нормировка гистограммы на величину norm:

Double_t scale= norm/h1->Integral(); h1->Scale(norm);

ГИСТОГРАММЫ

Вывод гистограммы
h1->Draw("options")
options – любые опции рисования, можно без пробелов (“sameCHIST”)

Некоторые варианты опций:

C — провести кривую через значения бинов;
P — нарисовать маркеры для каждого значения;
E — нарисовать погрешности значений (**E0-E6** – разные варианты изображения погрешностей гистограммы).

Чтобы нарисовать две гистограммы на одном рисунке, следует задать опцию same для второй:
h1->Draw()
h2->Draw("same")

Подробнее: https://root.cern/doc/v610/classTHistPainter.html#HP01

ГИСТОГРАММЫ

Нумерация бинов:

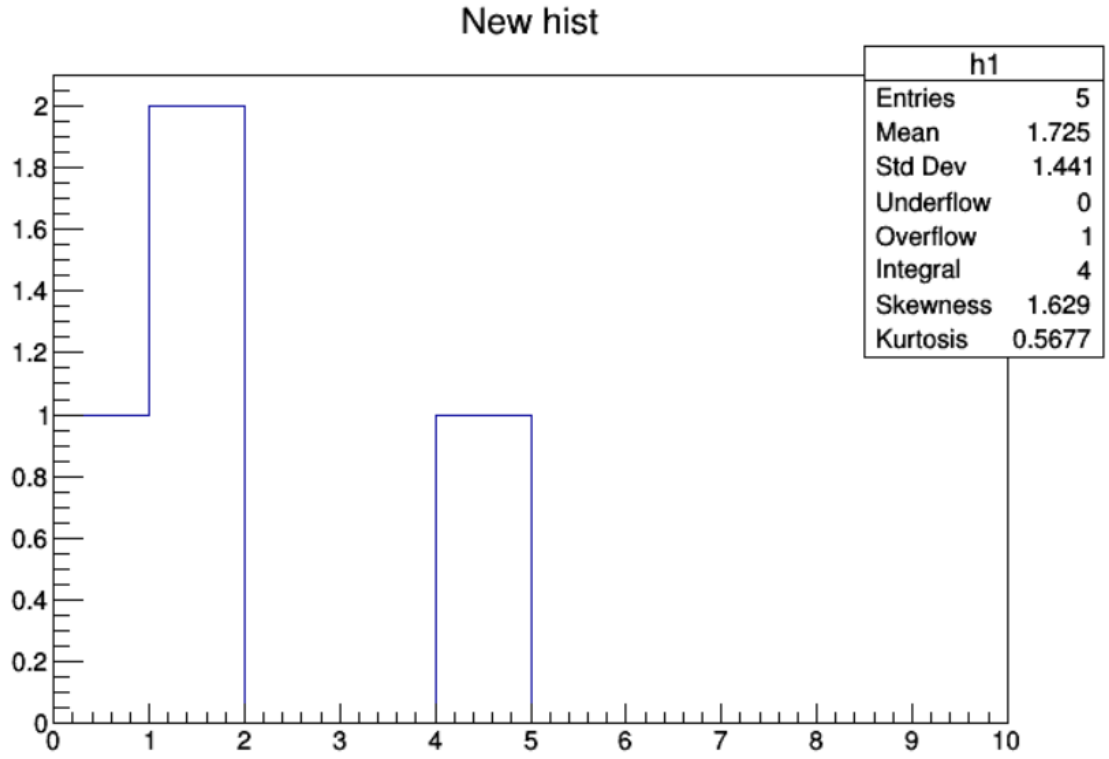
bin = 0; underflow bin (ниже диапазона значений)

bin = 1; первый бин ВКЛЮЧАЕТ нижнюю границу

bin = nbins; последний бин НЕ ВКЛЮЧАЕТ верхнюю границу

bin = nbins+1; overflow bin (выше диапазона значений)

```
TH1F *h1 = new TH1F("h1", "New hist",10,0,10);
h1->Fill(0);
h1->Fill(1.5);
h1->Fill(1.4);
h1->Fill(4);
h1->Fill(10);
h1->Draw();
```



ГИСТОГРАММЫ

Ошибки **non-weighted** гистограммы:

Рассматриваем случай одномерной гистограммы. Погрешность в каждом бине по умолчанию она вычисляется как корень из BinContent, где BinContent = h->GetBinContent(bin).

В случае, если гистограмма заполнена с весами равными 1 (weight=1), погрешность в каждом бине будет равна корню из числа вхождений в этом бине.

Погрешность интеграла гистограммы (без учета ширины бина) вычисляется как корень из суммы квадратов погрешностей каждого из бинов и может быть получена с помощью функции

integral = h->IntegralAndError(1, h->GetNbinsX(), err).

Значение погрешности будет возвращено в переменную err. В случае весов равных 1, погрешность интеграла гистограммы будет равна корню из числа вхождений в гистограмму:

h->GetEntries()

Ошибки для **weighted** гистограммы считаются по-другому.

ГИСТОГРАММЫ

В случае весов отличных от 1 вычисление погрешности бина как корень из BinContent неверно с точки зрения мат. статистики. Погрешность должна вычисляться как корень из суммы квадратов весов, как для каждого бина, так и для интеграла гистограммы.

В ROOT это реализовано возможностью записи суммы квадратов весов в каждом бине в массив fSumw2.fArray[1].
В случае, если существует этот массив, погрешность в бине гистограммы вычисляется как корень из суммы квадратов весов.

Начиная с ROOT v6, это делается автоматически, если при заполнении гистограммы хотя бы один[2] из весов будет отличен от 1.

В ROOT v5 и ниже, пользователь обязан сам включить запись квадратов весов, если они отличны от 1.

Есть два способа сделать это:

1.до создания гистограмм нужно вызвать функцию TH1::SetDefaultSumw2(kTRUE), тогда для любой созданной далее гистограммы будет автоматически создаваться и записываться fSumw2.fArray;
2.после инициализации гистограммы до начала её заполнения гистограммы вызывать функцию h->Sumw2().

[1] У пользователя нет прямого доступа к этому массиву.
[2] Заполненные до этого момента веса (если они есть) будут учтены в массиве fSumw2.fArray как BinContent в квадрате.

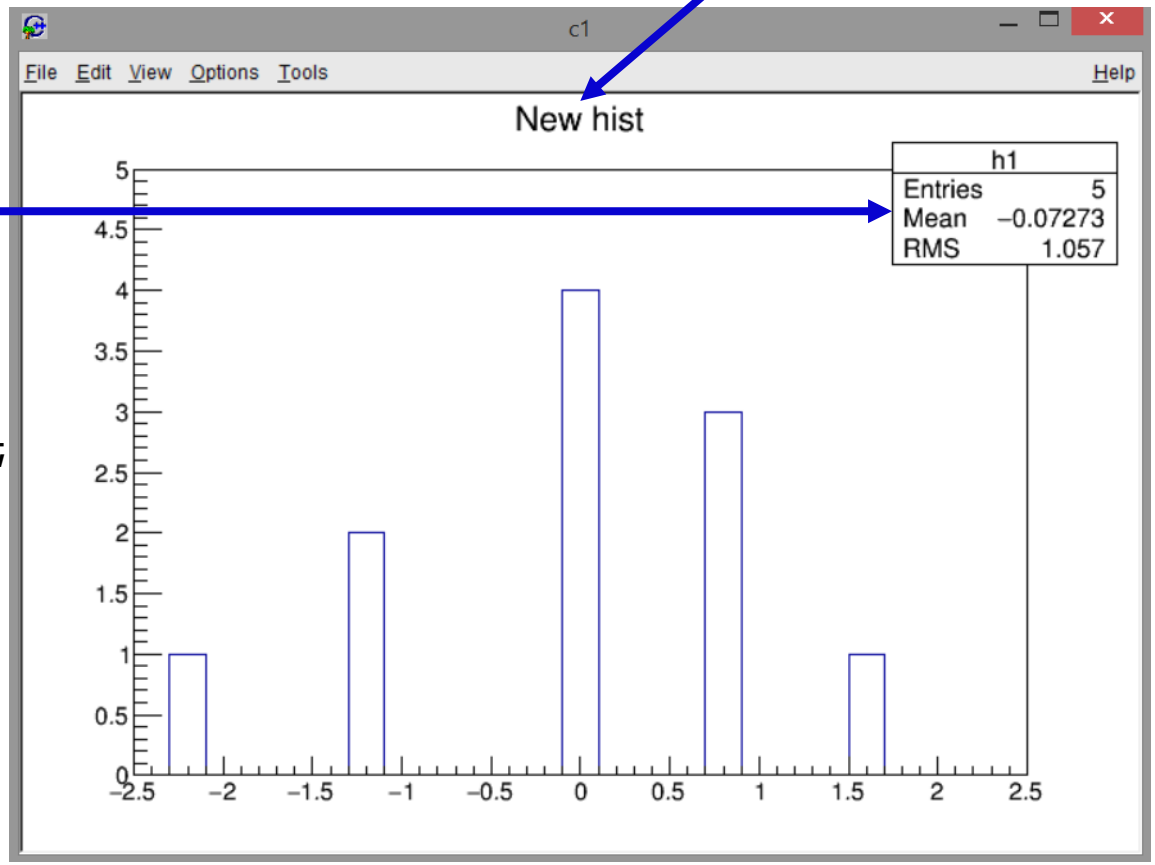
ГИСТОГРАММЫ

- Создадим, заполним и выведем гистограмму:

```
TH1F *h1 = new TH1F("h1", "New hist",25,-2.5,2.5);
h1->Fill(-2.3);
h1->Fill(1.5);
h1->Fill(0,4);
h1->Fill(-1.2,2);
h1->Fill(0.8,3);
h1->Draw();
```

Заголовок гистограммы

Окно стат. информации:
h1 – имя объекта;
Entries - число вхождений;
Mean - среднее значение;
RMS - оценка стандартного отклонения.
С помощью параметра SetOptStat можно увеличить количество выводимой информации в этом окне



ГИСТОГРАММЫ

- Методы Add(), Divide() и Multiply() позволяют складывать, делить и умножать гистограммы.
- Чтобы добавить к гистограмме h1 гистограмму h2

```
h1->Add(h2)
```

- Аналог, который не заменяет содержимое 1 из гистограмм:

```
h3->Add(h1, h2, c1, c2) // this = c1*h1 + c2*h2
```

h1, h2 – гистограммы, c1 и c2 – нормировочные коэффициенты.

- Можно добавить гистограмму с некоторым весом w

```
h1->Add(h2,w)
```

- Вычесть одну гистограмму из другой

```
h1->Add(h2,-1)
```

Складывать, делить и умножать можно только гистограммы с одинаковым числом бинов!

- Деление и умножение осуществляется аналогично:

```
h1->Divide(h2)
```

- Аналог, который не заменяет содержимое 1 из гистограмм:

```
h3->Divide(h1, h2, c1, c2) // this = c1*h1/(c2*h2)
h1->Multiply(h2)
h3->Multiply(h1, h2, c1, c2) // this = (c1*h1)*(c2*h2)
```

- Создать идентичную копию (клон) гистограммы:

```
TH1F *h1_clone = (TH1F*)h1->Clone()
```

ГИСТОГРАММЫ: ЗАДАЧА

Создайте 3 гистограммы с переменным биннингом.
Каждую заполните 5-ю произвольными значениями (с весами).
Далее сделайте и изобразите с ошибками 4-ую гистограмму, равную гист1*(гист2-гист3).

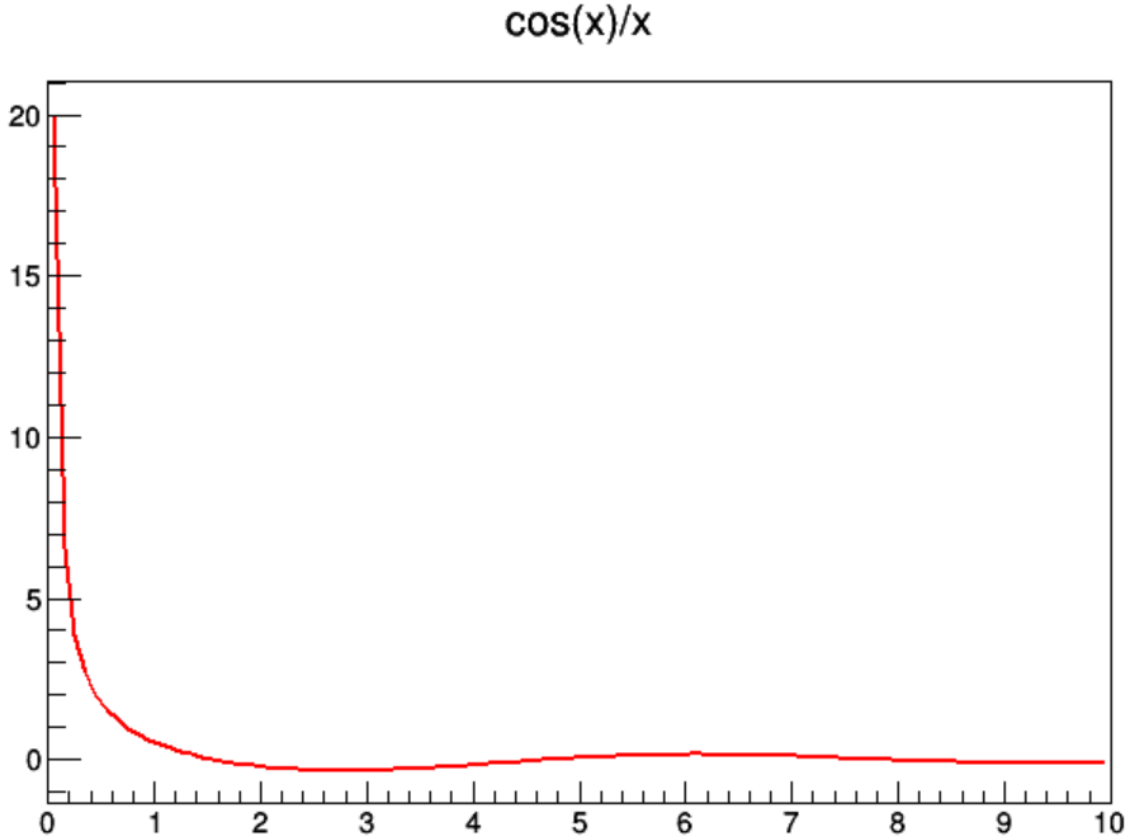
ФУНКЦИИ

```
root [0] TF1 *f1 = new TF1("f1","cos(x)/x",0.,10.)
(TF1 *) 0xc601c58
root [1] f1->Draw()
```

“f1” – идентификатор или имя функции

“cos(x)/x” – определяет вид функции

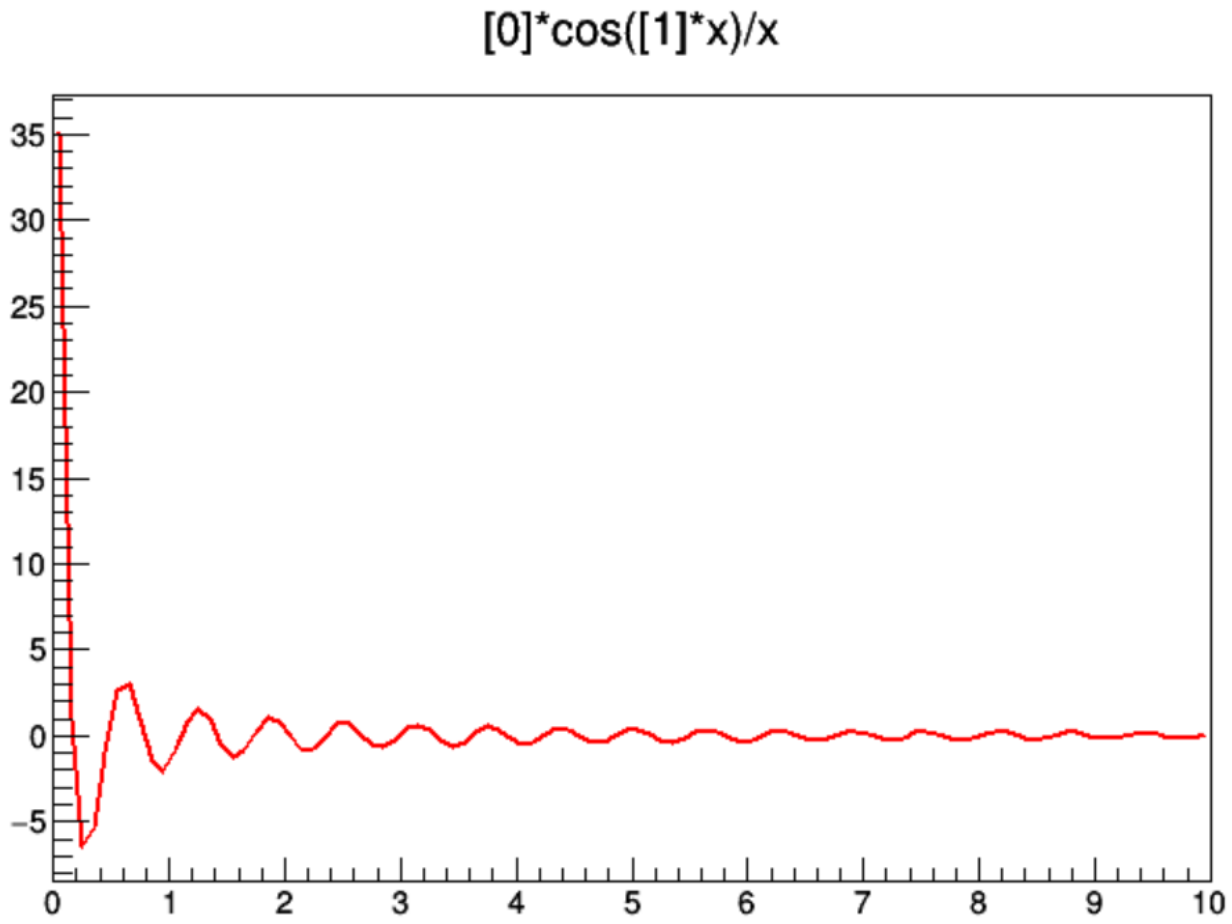
0., 10. – диапазон значений функции



Подробнее: https://root.cern.ch/doc/master/classTF1.html

ФУНКЦИИ С ПАРАМЕТРАМИ

```
root [0] TF1 *f2 = new TF1("f2","[0]*cos([1]*x)/x",0.,10.)
(TF1 *) 0xd2e12d8
root [1] f2->SetParameter(0,2)
root [2] f2->SetParameter(1,10)
root [3] f2->Draw()
```



ОБЛАСТЬ РИСОВАНИЯ «КАНВАС» (ХОЛСТ)

- Область, на которую выводятся графические объекты в ROOT, называется canvas (класс TCanvas):

```
TCanvas *c1 = new TCanvas ("c1", "canvas_new", 0.,0.,600,600);
```

Параметрами являются: название канваса, заголовок канваса и его размеры (координаты левой верхней и правой нижней точек). Вместо координат можно использовать просто параметр 1 и размер канваса будет автоматически задан.

- Объект, например, гистограмма, изображается на текущем активном canvas.
- Если canvas не существует, то он создается автоматически и имеет по умолчанию имя c1.
- Чтобы разделить canvas на несколько частей, можно воспользоваться методом Divide(k,l), где k и l число разбиений по горизонтали и вертикали соответственно.

- Чтобы выбрать, на какой части canvas'а следует рисовать объект, следует применить метод cd(n), где n — номер части. Разделы нумеруются слева направо, сверху вниз.

Ref. manual: https://root.cern.ch/doc/master/classTCanvas.html

TPAD (ХОЛСТЫ НА ХОЛСТЕ)

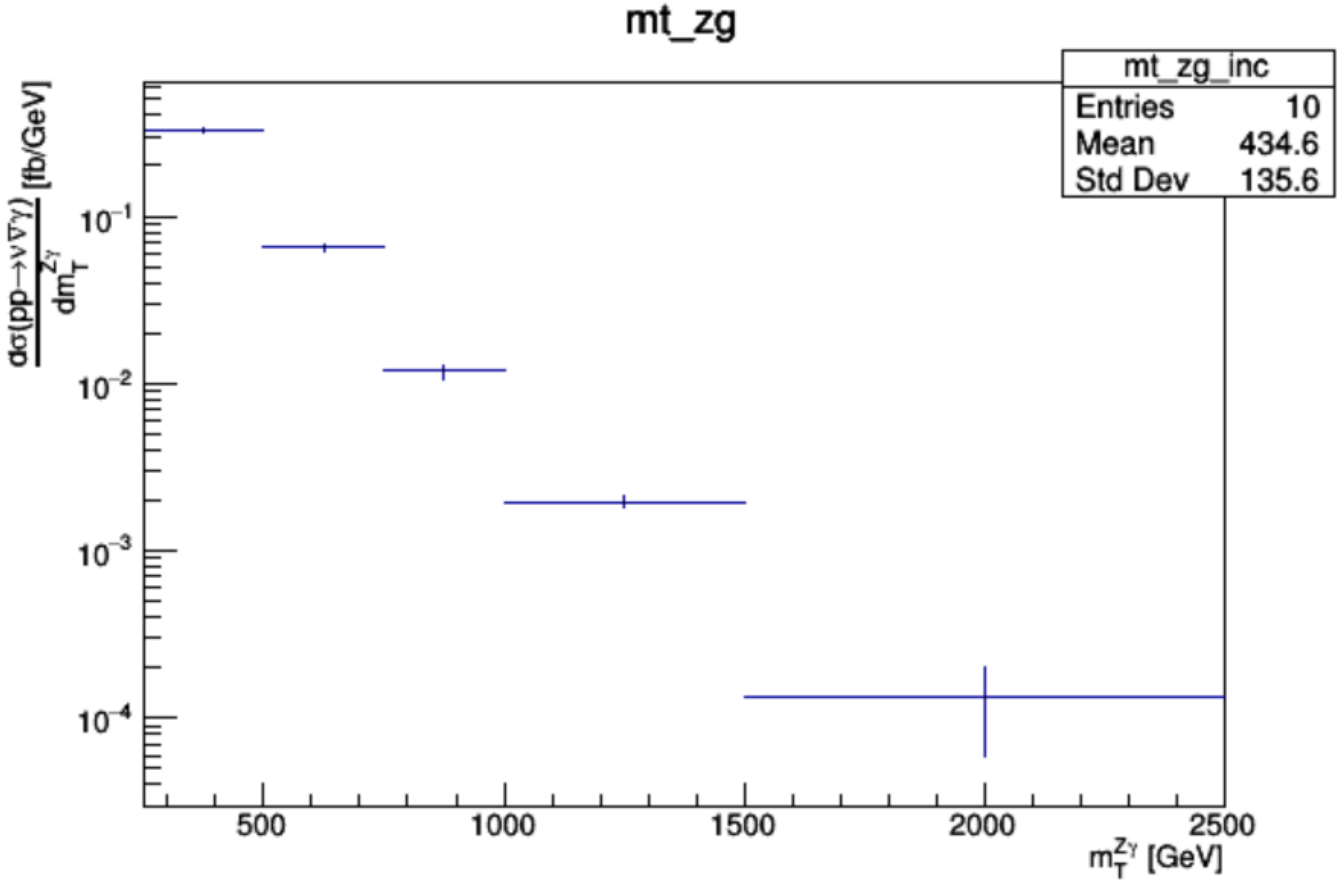
Канвас – это само окно рисование, автоматически область рисования, объект TPad, имеет то же имя.
Класс TPad позволяет создавать «саб-канвасы» (холсты на холсте).

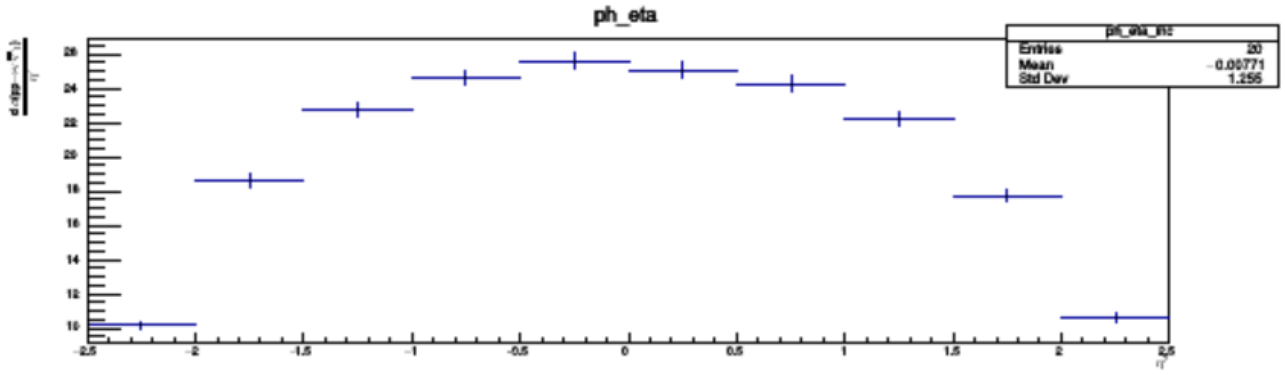
Когда мы делим канвас на части с помощью метода Divide(), то канвас делится на несколько равных по размеру TPad с именами имя_объекта_канваса_N, где N – номер TPad'а.
Если это делать не автоматически, то можно задать размеры для каждого саб-канваса.

```
TPad *pad_1, *pad_2;
TCanvas *c9 = new TCanvas ("c9", "mT_zg inc", 0.,0.,600,600);
pad_1 = new TPad("pad_1","This is pad_1",0.01,0.31,1,1.0);
pad_2 = new TPad("pad_2","This is pad_2",0.01,0.01,1,0.3);
pad_1->Draw();
pad_2->Draw();
pad_1->cd(0);
pad_1->SetLogy();
hist_mt_zg->Draw();
c9->Update();
pad_2->cd(0);
hist_ph_eta->Draw();
```


Ref.manual: https://root.cern.ch/doc/master/classTPad.html

TPAD (ХОЛСТЫ НА ХОЛСТЕ)





РАБОТА С ФАЙЛАМИ

- Чтобы сохранить гистограмму (или любые другие объекты) в файл, нужно прежде всего, создать файл (или открыть существующий)

- Работу с файлами обеспечивает класс **TFile**

Создание файла (объекта класса **TFile**):

TFile *f = new TFile("filehist.root", "new")

filehist.root - имя файла,

new или **create** - создать файл; если файл с таким именем уже существует, он не будет открыт

recreate - создать файл; если файл с таким именем уже существует, он будет перезаписан

update - открыть файл для записи; если файла с таким именем не существует, он будет создан

read - открыть файл для чтения (по умолчанию).

Root файл – бинарный файл, позволяющий хранить объекты различных классов Root. Позволяет использовать различные алгоритмы компрессии, что позволяет хранить и обрабатывать с помощью них большие объёмы данных.

Ref. manual: https://root.cern.ch/doc/master/classTFile.html

РАБОТА С ФАЙЛАМИ

- Файл после создания становится текущей ROOT-директорией

- Изначальная текущая директория - сессия ROOT
- Последний созданный файл есть текущая директория

- Глобальная переменная, указывающая на текущую директорию
  gDirectory
- Показать текущую директорию
  gDirectory->pwd() или .pwd
- Показать содержимое текущей директории
  gDirectory->ls() или .ls

Сменить директорию cd()

Закрыть файл Close()

РАБОТА С ФАЙЛАМИ

- Пример

```
root [0] .pwd
Current directory: Rint:/
Current style:     Modern
root [1] TFile *file1=new TFile("file1.root","recreate")
(TFile *) 0xb828a30
root [2] .pwd
Current directory: file1.root:/
Current style:     Modern
root [3] TFile *file2=new TFile("file2.root","recreate")
(TFile *) 0xb7a7348
root [4] .pwd
Current directory: file2.root:/
Current style:     Modern
root [5] file1->cd()
(bool) true
root [6] .pwd
Current directory: file1.root:/
Current style:     Modern
root [7] file1->Close()
root [8] .pwd
Current directory: Rint:/
Current style:     Modern
```

ЗАПИСЬ В ФАЙЛ

- Чтобы осуществить запись гистограммы в файл, используется метод Write()

- Пример записи гистограммы:

```
TFile *file = new TFile("filehist1.root", "new");
TH1F *h1 = new TH1F("hgaus","hist from a gaussian",100,-3,3);
h1->FillRandom("gaus",10000);
h1->Write();
```

- Чтобы записать все объекты в текущей директории (созданные после создания файла)

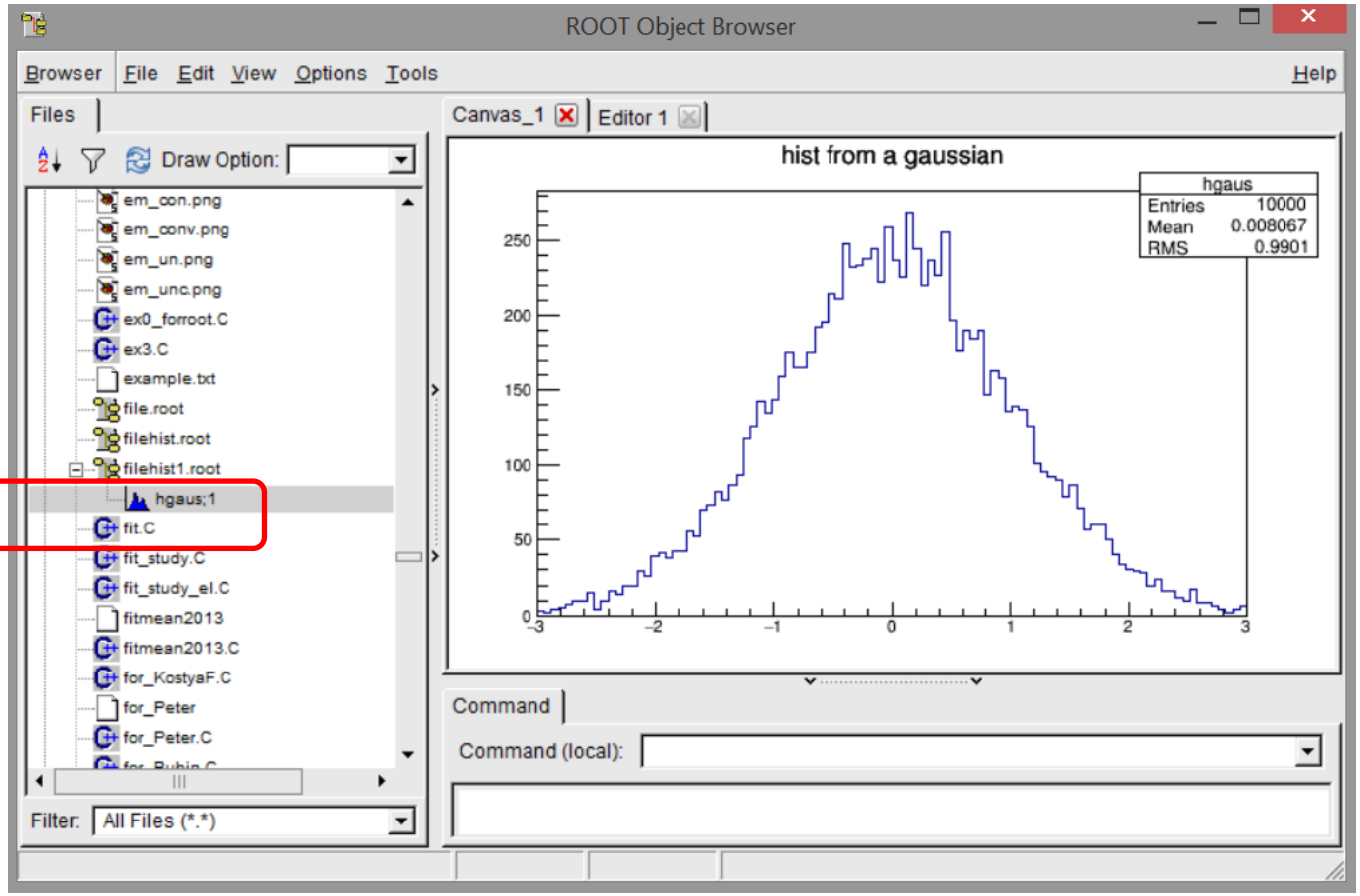
```
file->Write()
```

Если же надо записать объект, созданный до создания файла, то его надо записывать отдельно с помощью метода Write().

ПРОСМОТР ФАЙЛА В TBrowser

- Просмотреть содержимое .root файлов и объектов, записанных в них используется специальный браузер TBrowser.
- Чтобы запустить TBrowser, создайте объект этого типа в интерактивной сессии ROOT:

```
root [0] TBrowser b
```



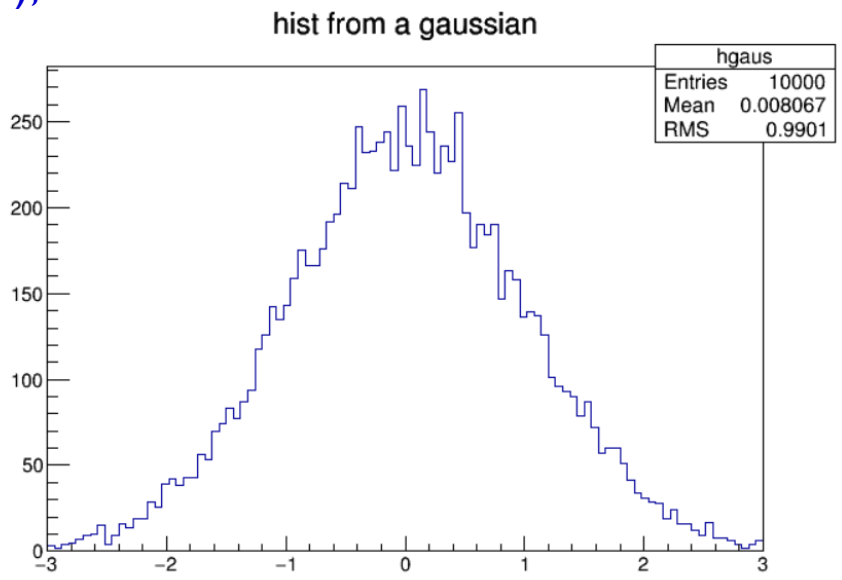
Ref. manual: https://root.cern.ch/doc/master/classTBrowser.html

ЧТЕНИЕ ФАЙЛА

- Чтение гистограммы (объекта):
- Метод Get("name")возвращает указатель на объект с именем name
- Пример:

```
// открываем файл для чтения
TFile *file = new TFile("filehist1.root");
// берем указатель на hgaus (идентификатор объекта, который мы видим в TBrowser
TH1F *h1 = (TH1F*)file->Get("hgaus");
// Рисуем гистограмму
h1->Draw();
```

В коде необходимо сделать **cast** к нужному типу.



ROOT-ФАЙЛ

Что происходит, если попытаться открыть файл, которого не существует?
Root помечает для себя этот файл как «зомби».
Если хочется вместо получения ошибки обрабатывать данную ситуацию в программе, это делается так:

```
TFile* file = new TFile("no_file.root");
if(file->IsZombie()) {
std::cerr << "Error: File is a Zombie! << std::endl;
```

ГЕНЕРАТОР СЛУЧАЙНЫХ ЧИСЕЛ

TRandom является базовым классом для генераторов случайных чисел в ROOT.

Интерфейс случайных чисел создан на основе производных классов TRandom1, TRandom2, TRandom3

Последний рекомендуется использовать для большинства задач из-за лучших характеристик: высокая производительность, период порядка 10^{6000}.

```
root [5] TRandom3 *r = new TRandom3()
(TRandom3 *) 0x10dd8328
root [6] r->Rndm(1)
(double) 0.99974175
root [7] r->Rndm(1)
(double) 0.16290988
```

Генерация происходит по равномерному закону.

Можно делать разыгрывание по любому теоретическому закону.

Например, через собственную функцию:

```
root [7] TF1 *f3 = new TF1("f3","abs(cos(x)/x)",0.,10.)
(TF1 *) 0xd522d48
root [8] f3->GetRandom()
```

Подробнее: https://root.cern.ch/doc/master/classTRandom.html

ГИСТОГРАММЫ, ФУНКЦИИ: ЗАДАЧА

Опять создайте 3 гистограммы, но теперь с равномерным бинингом.
Первую заполните случайным образом по экспоненциальному закону с параметром: 0.05.
Вторую – гауссом с параметрами mu=0, sigma=1.
Третью – равномерным распределением.
Далее сделайте и изобразите с ошибками 4-ую гистограмму, равную гист3*(гист1+гист2).

Запишите получившуюся гистограмму в файл формата root.